HELDA
HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

UNIVERSITY OF HELSINKI

https://helda.helsinki.fi

Energy efficiency of dynamic management of virtual cluster with heterogeneous hardware

Kommeri, Jukka

2017-05

Kommeri , J , Niemi , T & Nurminen , J K 2017 , ' Energy efficiency of dynamic management of virtual cluster with heterogeneous hardware ' , Journal of Supercomputing , vol. 73 , no. 5 , pp. 1978-2000 . https://doi.org/10.1007/s11227-016-1899-0 , https://doi.org/10.1007/s11227-016-1899-0

http://hdl.handle.net/10138/227977
https://doi.org/10.1007/s11227-016-1899-0

acceptedVersion

Downloaded from Helda, University of Helsinki institutional repository.

This is an electronic reprint of the original article.

This reprint may differ from the original in pagination and typographic detail.

Please cite the original version.

Noname manuscript No.
(will be inserted by the editor)

Energy efficiency of dynamic management of virtual cluster with heterogeneous hardware

Jukka Kommeri · Tapio Niemi · Jukka K. Nurminen

the date of receipt and acceptance should be inserted later

Abstract Cloud computing is an essential part of today's computing world. Continuously increasing amount of computation with varying resource requirements is placed in large data centers. The variation among computing tasks, both in their resource requirements and time of processing, makes it possible to optimize the usage of physical hardware by applying cloud technologies.

In this work, we develop a prototype system for load based management of virtual machines in an OpenStack computing cluster. Our prototype is based on an idea of 'packing' idle virtual machines into special park servers optimized for this purpose. We evaluate the method by running real high energy physics analysis software in an OpenStack test cluster and by simulating the same principle using the Cloudsim simulator software. The results show a clear improvement, 9% - 48% , in the total energy efficiency when using our method together with resource overbooking and heterogeneous hardware.

Keywords energy efficiency, OpenStack, Cloudsim, over-commit, heterogeneous hardware

1 Introduction

Outsourcing of computation has become more appealing and organizations are optimizing their IT costs by adopting cloud technologies. Besides the com-

Jukka Kommeri
Helsinki Institute of Physics,
E-mail: jukka.kommeri@cern.ch

Tapio Niemi
Helsinki Institute of Physics,
E-mail: tapio.niemi@cern.ch

Jukka K. Nurminen
Aalto University,
E-mail: jukka.k.nurminen@aalto.fi

mercial cloud providers, many academical cloud services have appeared, too. Amazon being the driving force in this development and its de facto standard API has been used in many other cloud frameworks [35, 47, 25]. Today, there exists several open source cloud-computing projects, that implement this de facto standard. The appearance of this new model of computing has made companies reconsider their IT processes and it also has enabled new companies starting Internet based services without the initial hardware costs. As several companies with different load patterns purchase their IT services from the same cloud provider and the cloud resources are located in the same data center, there is an obvious possibility to optimize the computing systems and increase their energy efficiency.

The basic function of the cloud computing framework is to enable the users to create and destruct virtual machines (VM) in a cluster of physical machines. These virtual machines can be created in different sizes and placed on physical machines, hypervisors, having the required resources; CPU, memory, network, etc. Using a virtualized system for consolidating IT services, such as the cloud framework, is already more energy-efficient than having just a single service on each physical server as it has traditionally been done in computing centers. The consolidation of VMs alone would be enough, if the workload of the virtual machines were constant, but this hardly is the case. Workload changes as a function of time and so the initially optimal placement of VMs may no longer be optimal. Thus, the location of virtual machine in a physical server needs to be changed.

OpenStack is one of many Amazon EC2 compatible open source cloud computing software frameworks. OpenStack is widely used and has a large community behind it. That is why CERN has also chosen it as a future platform for particle physics computing at CERN [29], [28]. Like many other cloud computing projects, OpenStack supports the basic features for creating, pausing, and destroying virtual machines. Additionally, it provides a way to spread virtual machines across a cluster of computers [23]. But like many other similar systems, it is missing a load based dynamic virtual machine management system for minimizing the energy consumption.

Dynamic placement of virtual machines have received a lot of attention among cloud computing researchers and practitioners during recent years (for example, [43, 38]). In practice, this means using a system that reacts to changes in virtual machine workloads in near real time by moving VMs between physical machines. The basic idea is the same as with static virtual machine management: to fit the virtual machines in physical resources in a way that is either power efficient or energy efficient, or otherwise allows the virtual resources to run in an optimal way.

We have previously found that virtual machines have very small impact on total load and power consumption when they are idle [24]. The less load a single virtual machine has, a single physical server can handle the more of them. In this paper we introduce a load balancing system that tries to maximize the load on computing nodes and minimize the amount of active computing nodes. We achieve this with dedicated hosts for storing idle virtual

machines, active monitoring of resources, and load based active management of virtual machines in the physical cluster.

In our work, we take both the system throughput and energy efficiency into consideration by utilizing heterogeneous hardware in a dynamically managed virtual machine cluster. We develop a system for virtual machine management for OpenStack and introduce a new way of packing idle virtual machines into dedicated park servers. By 'idle', we mean that the virtual machine has little or no workload to run at the moment and we define it by using the load value of the operating system. We test the system with high energy physics workload and also verify the algorithm with a simulator. We show that energy efficiency of a cloud computing cluster can be improved by using dynamic management of virtual machines and prioritizing on energy-efficient hardware on the heterogeneous cluster.

We have structured the rest of the paper as follows. First, in Section 2 we cover basics of cloud computing and existing cloud management systems. In Section 3, we introduce our load balancing algorithm. Next, in Section 4, we describe the test software and test environment used in this study, which is followed by the results given in Section 5. Finally, we end with conclusion in Section 6.

2 Related work

The adoptation of virtualization has resulted in a plethora of both commercial and open source virtual machine management tools. At first, these tools were mainly made to control single hypervisor systems or to help system administrators to manage their hypervisor clusters. Cloud management frameworks making Infrastructure as a Service (IaaS)-type services possible, were the next step in this evolution. They provide, on top of the basic hypervisor management, a large collection of user management and monitoring tools.

2.1 Cloud management systems

Cluster management systems are often based on heuristics that use data gathered from both physical and virtual machines. In the simplest form, these systems make decisions based on the load of physical CPUs, while the complex ones take into consideration additional aspects such as memory usage, network traffic, service level agreements (SLA), server energy consumption, server thermal state, virtual machine intercommunication etc. Even the simplest cluster management algorithms improve energy efficiency of a cluster that has a varying workload [5].

The basic idea of these management systems is to maximize the usage of an active server and minimize the amount of them. One way to achieve this is to pack already loaded servers more efficiently. Cloud management systems can roughly be categorized into active and passive systems. As mentioned earlier, OpenStack, like many other open source systems, do passive management

where load balancing is done at the time of creation of new virtual machines. Even though these algorithms can balance load between physical hosts, they are limited in reaction to the load changes of instances. Active cloud management systems can better react to changing load patterns as they actively move the virtual machines between physical machines to meet the goals of the system. These goals can be energy efficiency, QoS, SLA, etc. Active systems usually react to several events indicating that the system state is not optimal or some thresholds have been breached.

Many authors have studied virtual machine management, i.e. consolidation, and they have proposed several algorithms for solving the resource optimization problem. These algorithms vary much in complexity and therefore their comparison is difficult. As Srikantaiah et al. [41] note that development of an algorithm is a compromise between time and performance. The algorithms that produce the best results tend to require more time for calculations, e.g., the constraint programming algorithm by Hermier et al. [20]. However, even the simplest algorithms, such as the one by Shrikantaiah [41] that picks the most suitable migration target by computing a simple euclidean distance calculation, can improve the energy efficiency of a cluster as then the system reacts to the change of resource requirements. Like other algorithms based on distance metrics, the euclidean distance algorithm assumes that the resource requirements of the new task are known, which is not the case with many real world workloads.

Many of the proposed cloud management systems use various bin packing algorithms and are centrally managed [20], [45],[40], [31], [10], [26]. Verma et al. [45] introduce pMapper cluster management system. Like many other other centrally managed system, pMapper has a separate monitoring service to get the latest hypervisor resource usage and the load of different virtual machines. Based on this knowledge of the state and service constraints, a new virtual machine placement is formed. Prerequisite for the functioning of pMapper is forming of power models for different hardware. PMapper optimizes energy efficiency of a computing cluster by organizing virtual machines by their power model. Virtual machines are placed using the first fit decreasing (FFD) bin packing algorithm. The algorithm places virtual machines on servers where their energy increase is minimal. Migration cost is only calculated from the memory allocation of the virtual machine at the beginning of the migration ignoring the overhead introduced by the workload memory dirtying. This can be quite inaccurate if the virtual machine is active. Hermier et al. [20], with their Entropy consolidation manager, divide virtual machine management in a computing cluster into two sub-problems; 1) virtual machine packing problem, and 2) virtual machine replacement problem. In the first phase a minimalist configuration is determined as a constraint satisfaction problem (CSP) where VM resource requirements (CPU and memory), are mapped to physical resources. They use a dynamic programming approach to solve this multi knapsack problem. As the solution space is vast, the algorithm is given a reasonable lower bound, the amount of physical nodes, and an upper bound with FFD. In this way they can get a more optimized configuration in less time. In the second

phase, the transition from the previous configuration to this new minimized configuration is also considered as a CSP. A reconfiguration plan takes into account migration cost and also possible linked migrations and temporary migrations, e.g. VMs can be moved temporarily to another host or moved to their destinations through intermediate hosts. The algorithm takes a long time to complete, but compared to FFD, it provides some improvement to VM packing efficiency. Again, migration cost is calculated inaccurately directly from VM memory allocation.

The previously mentioned active management systems have been centralized. Centralized systems require complex management and are more vulnerable to failures than decentralized systems. The more machines there are, the harder the optimization calculation becomes. To be able to scale, a decentralized solution is needed. A decentralized version by Wang et al. spread the load status of the cluster to every node with load vectors [46]. Physical nodes communicate their state to random peers on regular intervals. These status updates are stored into vectors and used when choosing a destination for virtual machine migration. A migration decision is made at the time of the status exchange if the load of a physical node either passes an upper threshold or is less than a lower threshold. The virtual machine to be migrated is chosen such that the migrations will cause the source system load to be between the lower and upper thresholds. This is a simple solution that only considers CPU utilization. For example, it omits the migration costs. Wuhib et al. have made a similar system as Wang et al. on top of OpenStack using a gossip protocol [48]. Every node keeps track of host and virtual machine load changes and shares a moving average load with the other hosts using the gossip protocol. The gossip protocol also chooses a random sample of hosts for information sharing. The algorithm is run on intervals and also as a result of a load balancing event.

There are also other studies such as Pahlavan et al. [34] who studied power reduction in data centers by developing new server placement methods for chassis consolidation, i.e. placing active servers on the same racks. The authors see this as a similar approach as using virtualization technologies but it avoids the virtualization overhead. Our method presented in this paper can be combined with this approach. Finally, Hosseinimotlagh et al. [21] developed new energy-aware scheduling methods for real-time cloud computing. The idea is to run each host server on the energy-optimal level of workload. Our method uses the same principle when making decisions on VM migrations.

As we can see, a lot of research has focused on amidst the dynamic management of a virtual computing cluster. All algorithms aim at minimizing the need for physical servers. Some emphasize the optimal placement of virtual machines in the cluster and do this by performing complex optimization calculations. Some take into account migration cost at a very high level and loose accuracy. Head to head comparison of these algorithms is difficult. The choice of an algorithm depends on its application. Even the heaviest algorithms might turn to be useful in an environment where load does not vary very much, e.g. high-energy physics computing where single analysis tasks can last days and their resource usage does not fluctuate during this time (See Section 4.1.1). On

the other hand, outside supercomputing, we have web services that have very fluctuating resource requirements. Running these optimally in virtual machine cluster requires fast response from the management system.

2.2 Heterogeneous hardware in cloud computing

So far research on heterogeneous hardware in a cloud environment has mainly focused on non-homogeneous hardware in big cloud environments such as Amazon cloud [33], [18], [12], or to the use of specialized hardware such as GPUs in clouds [49]. Cloud environments can have different types and generations of processors [33]. This can cause the performance of virtual CPU in a virtual machine be different among virtual machines of the same type. Besides being unfair to clients, the difference in hardware can also hurt performance. For example, Gupta et al. [18] have found that the heterogeneous hardware of a cloud environment can be harmful for HPC load when it is stalled by its synchronous nature. As a solution, they suggest a method that picks homogeneous hardware from the pool of VMs.

In this study, we propose improving the total efficiency of the system with heterogeneous hardware and dynamic load migration. Some studies already point out that exploiting heterogeneous hardware in a virtualized cluster can be beneficial. Hirofuchi et al. [42] use dedicated servers for virtual machines with less load, shared servers, and dedicated servers for running heavily loaded virtual machines. In their system, virtual machines are moved between these two types of hardware as a function of their processing needs. Shared nodes would store idle virtual machines and dedicated nodes would run active virtual machines. When a shared node CPU load exceeds 90% the most active virtual machine is moved to a dedicated node. When the CPU activity of a virtual machine is less than 50% it is returned to a shared node. Dedicated nodes can have as many virtual machines as they have CPU cores and they are filled such that the dedicated node with smallest number of empty slots is filled first. In their work, the only difference between the shared node and the dedicated node is the amount of memory.

Profiting from energy-efficient hardware was also studied by Verma et al [45], but in their study the implementation was left at the level of power models where different hardware had different power models. Further, the preference of allocating virtual machines to more energy efficient hardware has been studied by Nathu et al. while making the VirtualPower system [31].

2.3 Over-committing resources

Over-committing and overbooking are terms used with virtualization when speaking about provisioning more resources to virtual machines than available on their physical hosts [44],[13]. Over-committing offers a solution to make better use of the existing hardware as the average utilization rates of virtual

machines can be a fraction of the requested [4], [16]. The risk is that when virtual resources are fully used, there is not enough physical resources to offer. We can mitigate this problem by having safety margins for the resource overbooking. The safety margin can be based on the history of virtual machine loads and taken into account when new virtual machines are created. But the margin can be wasteful if a certainty of the existence of resources is required. This issue was studied by Li et al. [26] when developing EnaCloud. They added an over-provisioning function to minimize the amount of migrations, but ended up using more energy. To improve the utilization rate further, forecasting the aggregate workload of the virtual machines could be used [16],[7].

3 Load balancing in heterogeneous clusters using migrations

New servers are becoming more efficient and energy-proportional, but their energy consumption range is still 20% to 100% of maximum energy consumption [9]. Furthermore, the energy-consumption of new energy-proportional hardware is not linearly related to load and the optimal load is between 40%-80%. There are also clear differences inside a processor family in energy consumption and 20% of the energy consumption of a powerful processor is much more than that of an energy-efficient end of the lineup [22]. Generally, idle running servers are not energy-efficient and also not very cost-efficient. It would be more cost efficient to pack the existing virtual machines better, i.e., place the idle VMs to a special hardware, and then shutdown the unnecessary hardware.

Computing clusters, in general, consist of a powerful front-end, special hardware for storage, and a homogeneous group of computing nodes. In the case of physics computing at CERN, the computing nodes are equipped with 2 GB of memory per computing core. Processors are usually from the mid-range of the current processor family, because they are powerful enough but not overly expensive. High-energy physics computing at CERN is performed using batch systems in a distributed way. The significance of a single node is minimal and in the case of node failure, its load can be re-run on another node.

In a batch system, like in CERN, physical resources are divided into slots. Every node has a fixed amount of slots and usually this number corresponds to the number of CPU cores. Workload, consisting of high-energy physics (HEP) analysis or simulation jobs, is distributed in a round robin way evenly into the cluster independent of what kind of load the other slots are running. This approach could have limitations related to memory management in NUMA architectures but in the case of the HEP computing it generally works well, since memory requirements of individual jobs are still quite small. Workload runs a long time and has phases where it uses less CPU, e.g., waiting for the data to be uploaded from the remote storage to the local storage. By remote storage we mean a data storage in a different datacenter and by local storage, a local storage system in the same datacenter. Since the job can analyse millions of events possibly stored in different files, we cannot always transfer all required

data before the job starts. This causes the physical node or at least the current slot to rest idle. This idle waiting could be done in a server with less computing power. The load is analogous to, for example, server-based computing systems where desktop sessions may run idle for a long time or web servers where the query rate fluctuates as a function of time of day. Although the load fluctuates, the servers need to be online and ready to answer.

In our earlier work we have found that virtual machines have very small impact on total load and power consumption when they are idle [24]. As it is in the case of high energy physics analysis, when a computing job is waiting for data and thus using very little CPU or memory resources on the physical host. Similarly the resources are unutilized when a virtual machine is waiting for the next job from a batch system. This waiting time could be spent on an energy-efficient server, a server dedicated for parking idle virtual machines, but still capable of network transfers. The less load a single virtual machine has, a single physical server can handle the more of them.

The basic idea of our load balancer is to make space for active instances on computing nodes, maximize the load on computing nodes and minimize the amount of active computing nodes. This is achieved with specialized hosts for storing idle virtual machines, active monitoring of resources, and load based active management of virtual machines in the physical cluster.

In our cluster, we have two types of servers; park servers and computing nodes. The park server is used to store idle virtual machines and its over-commit ratio is high ,i.e. there can be multiple virtual machines per CPU core. Instead, computing nodes are used by active virtual machines and have one to one mapping of virtual and physical resources, i.e. no over-commitment is used.

Virtual machines are considered to be either active or idle. Idle would be a state where the virtual machine is waiting for work and in the active state it is processing a given workload. Depending on the activity of a virtual machine, it is placed either on an energy-efficient park server or on a computing node. Idle virtual machines have minimal need for physical resources and it is possible to over-commit physical resources to them without affecting their quality of service. They can be stored into park servers where they can wait for a more active state. This way the virtual machines, that require more processing capacity, can be served with more dedicated hardware and the virtual machines that are waiting for more work take less space as they can be packed more densely. Active virtual machines need more resources and the less they need to contend for physical resources the better the service level.

The load balance manager has three functions:

1. Move idle virtual machines to the park server.
2. Move a virtual machine from the park server to a computing node when an idle virtual machine becomes active.
3. Pack virtual machines within the cluster to minimum number of servers.

Function 1 moves idle virtual machines from computing nodes to the park server. Here the migration decision is based on the load of a virtual machine. If

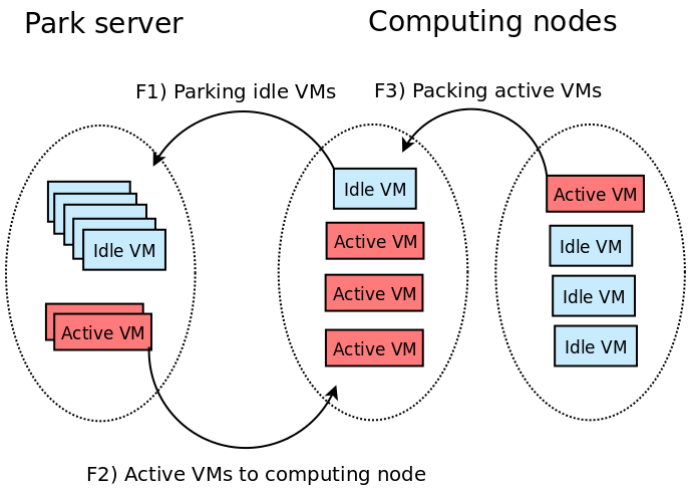

Fig. 1: Load balance functions

the load of virtual machine goes under a given threshold, the virtual machine is migrated. Function 2 does the opposite and moves loaded virtual machines from park server to computing nodes. It takes action if the load average of a park server surpasses its threshold. The virtual machine to move is chosen among the active virtual machines on the park server. Finally, Function 3 moves active virtual machines from underutilized computing nodes to better utilized computing nodes. It attempts to optimize the load on computing nodes and thus improve their energy efficiency.

4 Test environment

We evaluated and tested the functionality of our load balancing algorithm in a small OpenStack test cluster and the scalability of the solution by using the Cloudsim simulator software. Cloudsim was complemented with our load balancing algorithm and modified to support heterogeneous hardware.

4.1 OpenStack test cluster

OpenStack [11] is one of the open source cloud management systems that have gained popularity and is already used by many private and public companies and institutes [11]. It has a lot similarities with other open source cloud projects such as Eucalyptus and OpenNebula. They all support the Amazon APIs; EC2 and S3 and support the main virtualization technologies; Xen, KVM and VMWare for hypervisors through an interface, Libvirt [1].

[1] http://libvirt.org/

OpenStack is a collection of services, that operate hypervisors, handle virtual machine images, enable network connectivity between virtual machines between physical servers and many other user and resource management and monitoring utilities[2]. The required services needed to run virtual machines are called Nova, Glance and Keystone. Nova is responsible for the management of virtual machines, Glance provides virtual machine images and Keystone administers permission to different services. Other services are Horizon that implements a graphical user interface and Cinder that provides permanent storage for virtual machines. Without permanent storage, virtual machines or instances, which they are called in cloud environment, do not save any state when destroyed. Although, Nova comes with a network service, Neutron service extends it by providing more fine-grained control over the network configuration and also providing firewall and virtual private network creation. Our test environment uses an earlier version of OpenStack, Havanne.

The Nova scheduler handles resource balancing in OpenStack. The Nova scheduler comes with 3 different algorithms: a simple, random, and filter scheduler [27], but it also provides a possibility to add new schedulers as plugins. OpenStack does not include a dynamic load based virtual machine manager that would migrate virtual machines between physical hosts. Our load balancer extends Nova schedulers functionality and provides an active load based manager, that commands Nova with Openstack API.

4.1.1 Test setup

We developed a centralized monitoring system for our active management tests. This system stores information both from virtual machines and physical servers. It uses the information that is gathered by the Oracle Grid Engine (OGE) and additionally, CPU load, CPU utilization, and memory state are periodically submitted by hypervisors. All the information is stored in a relational database. Our load balancing algorithm retrieves the system state from the database through a web server with JSON.

In our tests we measure the energy efficiency of high-energy physics (HEP) analysis software in a dynamically managed cloud environment and compare it to standard static cloud. HEP computing used in the LHC experiment is both CPU and data intensive. It comprises mainly the analysis of large amounts of both generated and measured data. The generated data is produced with simulations and compared against the real data measured by the experiments at CERN. A single high-energy physics analysis can go through millions of events [36]. This work can be parallelized easily as the analysis of a single event does not depend on the analysis of another event. Normally, the analysis starts from 600 - 1,000 separate processes and each of these analyze a subset of input events. Physics events are stored in a database like container, ROOT files [2]. Input file sizes in the ROOT format are about 100 - 300 MB and each

[2] http://docs.openstack.org/juno/installguide/install/apt/content/ch_overview.html

file contains 700 - 2,000 events. An average analysis job would pass through 10 - 25 million events that are stored in 30,000 - 40,000 files.

For this purpose we set up an OpenStack installation using DevStack[3]. Workload was distributed to the cluster as a batch of jobs using OGE[4]. Twelve OpenStack t1.small virtual machines instances [17], worked as OGE execution nodes. Submission intervals of the batch jobs follow a Poisson process [37] and resemble the real distribution of job arrival rate in CERN clusters. Although the workload was real physics computation, its duration was artificially limited. The durations of analysis jobs were scaled down to allow more reasonable test times. Batch jobs consisted of 4 minute, jobA, and 7 minute, jobB, jobs in equal proportions.

Physical servers were installed with Ubuntu 13.10 and Devstack, a stateless OpenStack installation[5]. Virtual machines were installed with Scientific Linux at CERN 5 (SLC5), Cern virtual machine file system (CMVFS) [30] version 2.1.19 and OGE 6.2u5. CVMFS is a new way to bring CMS Software (CMSSW) [15] to the computing nodes. It mounts the software to a local filesystem from CERN servers and caches some of the used software locally as they are used. In our tests, a version 6.2.8 of CMSSW was used.

Our test cloud consisted of three servers with two different hardware compositions. As a park server and as an OpenStack front-end we used Dell 210 with Intel X3430 processor and 32GB of memory. As computing nodes we used the same servers with 12GB of memory. OpenStack servers were connected to an NFS share, that contained virtual machine images. Servers were connected using two gigabit networks. The first network was used for OpenStack communication and the second was dedicated for NFS. Power usage data of the servers was collected using a Watts up? PRO meter via a USB cable. Power usage values were recorded every second. A separate PC was used to control the tests and collect electricity measurements. The test setup is illustrated in Figure 2.

Thresholds for the load balancing functions described in Section 3 are based on the average load values of Linux. In the first function of our balancer, the migration decision is based on the 1-minute average load of the virtual machine. If the one minute average load of virtual machine goes under the given threshold, the virtual machine is migrated to the park server. The second function does the opposite and moves loaded virtual machines from the park server to computing nodes. It takes action if the one minute load average of the park server surpasses its threshold. In our tests, the thresholds of load were 0.5, minimum and 1, maximum.

[3] http://devstack.org/

[4] http://en.wikipedia.org/wiki/Oracle_Grid_Engine

[5] http://docs.openstack.org/developer/devstack/

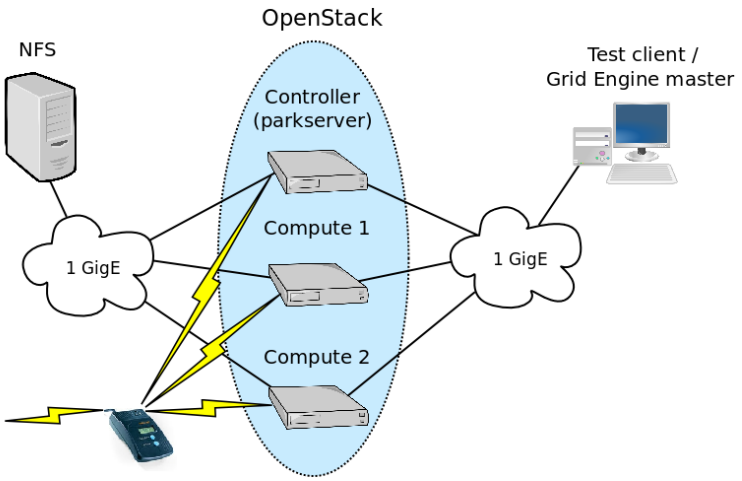

Fig. 2: Test environment setup

4.1.2 Tests on real hardware

In our test with real hardware, we compare basic OpenStack installation with passive management to our own active management algorithm. Tests compose of three different cases:

1. SP - Static placement of instances
2. AM - Active management of instances with heterogeneous hardware prioritization
3. AMP - Active management of instances with heterogeneous hardware prioritization and active consolidation of active VMs.

In Case 1, instances are created permanently for the whole duration of the test. In Cases 2 and 3, the placement of instances is static and virtual machines are migrated between physical machines depending on the load. Starting configuration in our test cases is slightly different. In the static placement scenario, virtual machines are spread equally into all nodes. In the other two cases, the virtual machines' initial placement is on the park server. All the measurements are repeated 10 times to get statistically comparable results.

4.2 Dynamic management simulation

In order to test our load balancer in a larger environment and to evaluate whether the results from our physical environment are also relevant in production-scale systems, a simulator was needed. Because of different aims of the tests, the simulated results cannot be directly compared to the test with real hardware. We decided to use an existing Java based simulator, Cloudsim, that has been widely used for simulating cloud environments [1]. Cloudsim

Type	Memory (GB)	Frequency (MHz)	Cores
Park server	16	2400,3400	2
Computing node	4	3400	4
VM	1	2000	1

Table 1: Simulator server set up

provided most of the necessary features to simulate dynamic management of cloud infrastructure and collect energy consumption information [8]. However, adding support for heterogenous hardware was needed to be able to simulate our test case. The load balancer, that was described earlier in Section 3, was also implemented into the simulator; it uses the existing algorithms of the simulator [6]. Also the separation between normal computing nodes and park servers were implemented.

The Cloudsim simulator was set up to run 60 virtual machines in 15 physical hosts. The physical hosts were divided into park servers and computing nodes. The number of park servers was varied in the tests, but the total server count remained constant. Every virtual machine had a separate input file that described their CPU load for every scheduling interval. In our tests, the scheduling interval was set to 30 seconds and the duration of one simulation to 24h.

The input of the virtual machines is periodical. Constant idle periods are separated with higher load periods that resemble real physics workload, a physics job. Some randomness was added to the beginning of the input so that the first high load period of a virtual machine would be in different spot on every virtual machine. Three input sets were created with different idle periods, such that we would have 50, 100 and 150 physics jobs of 8.5 minutes in one simulation. In every input set, the simulation duration was the same 24 hours, 2880 scheduling intervals. We created 5 variations of every test set in order to test the effect of the different random seeds. The input set randomness had little effect and the deviation between results with different random seeds was in the magnitude of one thousandth.

Two different power models were used. For the energy efficient server we used a model that mimics the one of Intel Xeon E3-1260L (2.4 GHz)[6] and for the normal server we used the one of Xeon E31280 (3.4 GHz)[7]. These power models define how much energy is consumed with different processor loads. Power models were obtained from SPEC website and they are the results of measurements with SPECpower_ssj 2008 software [8] on real hardware.

In Table 1, we have the hardware parameters of the simulation set up. As a park server, both hardware types were used in different tests to compare the effect of hardware on energy efficiency.

[6] "http://www.spec.org/power_ssj2008/results/res2011q2/power_ssj200820110531-00379.html"

[7] "http://www.spec.org/power_ssj2008/results/res2011q3/power_ssj200820110806-00393.html"

[8] https://www.spec.org/power_ssj2008/

5 Results

5.1 OpenStack cluster

The aim of the OpenStack test was to show that the load balancing method works in practice and migrations do not cause a significant increase for energy consumption or processing time. The energy saving potential of the method will be demonstrated by using simulations.

As assumed, active management of virtual machines does not come without a cost but the cost is reasonable. Figures 3 and 4 both show, that total energy consumption is higher and job durations slightly longer when the virtual machines are moved in the cluster. Figure 3 shows the energy consumption of three different test cases; SP test case where no active management is used, AM where the first two functions of our load balance manager is used and then AMP where the third, packing, function is added. We can see that the migrations do not introduce much overhead to the energy consumption or processing times. The energy consumption measurements do not take into account the benefit gained by shutting down the unnecessary servers. This benefit is presented in Table 2 at the end of this section.

We can also see that the energy efficiency increases with higher load values: if we triple the submission rate, energy consumption increases by only 12%. This is naturally explained by the relative high idle power of the computing servers.

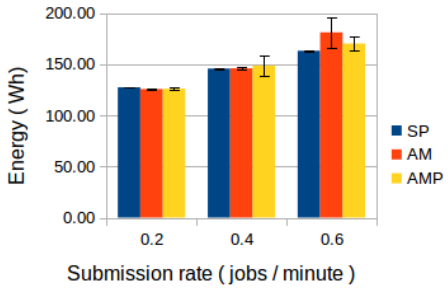

Fig. 3: Energy consumption of the test cluster with different loads and scheduling algorithms

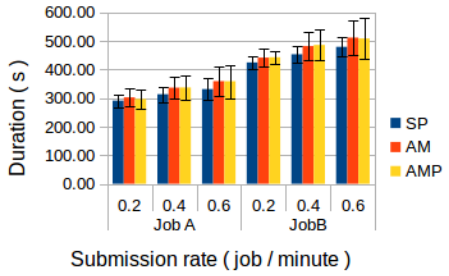

Fig. 4: Processing times of two different test jobs when running the same tests with different loads and scheduling algorithms

Improved energy efficiency usually comes with a cost as it is the case here as well. We can see this cost in the form of slightly higher processing times of physics jobs. In Figure 4, we have the run times of two different jobs, jobA and jobB with different submission rates. We can see that the static placement gives about 13% shorter times than the two active management cases. As we can

see in Figure 4, the active consolidation of VM instances (the AMP method) does not have a significant effect compared to the normal AM method.

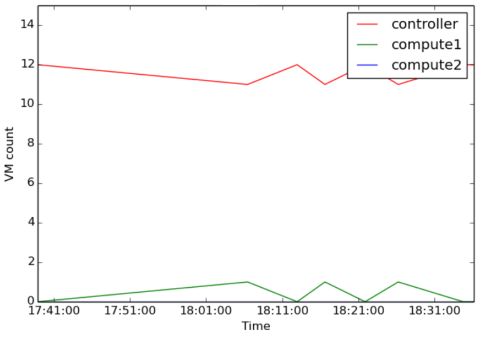

Fig. 5: VMs in the cluster during rate 0.2 test

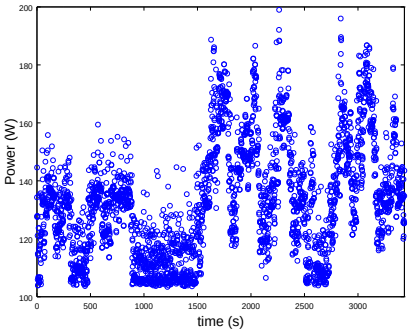

Fig. 6: Cluster power consumption during the rate 0.2 test

To illustrate movements of virtual machines among the physical servers during different tests scenarios, we picked samples of migration data using the tests without the active VM consolidation feature, i.e., the AM tests. Figures 5, 7 and 9 illustrate the movements of virtual machines between the three physical servers. In Figure 5, we have the movements of one test with the rate 0.2, Figure 7 with the rate 0.4, and Figure 9 with the rate 0.6. Rates 0.2 and 0.4 are very low and two servers they can basically serve them. Based on the tests, we can indicate that the AM and AMP functions are able to increase the utilization of servers compared to the static placement of instances (the SP method). On the right side of these figures we have the power consumption of the corresponding tests in Figures 6, 8, and 10.

Since the AM scheduler without packing does not attempt to pack load, i.e. consolidate VMs into smaller number of computing nodes, this results in the physical servers running with low utilization. As we can see in Figure 7, the computing node two runs a long time with just one virtual machine. These under committed resources are corrected with the use of packing function.

Rate 0.6 is high enough to saturate the used test cluster. When this happens, the park server load is always moved to the most loaded computing node that still has enough space left to support the additional load. Computing nodes are allowed to host only four virtual machines in order to guarantee the one to one mapping of CPU resources. The park server, on the other hand, can host more virtual machines.

A host power management system was not implemented into this cluster, but we have previously shown that it is possible and beneficial for a batch cluster [39]. Instead of making a similar solution, we have measured the idle times of the test servers to show the benefits of active management of virtual machines. In order to work, the controller of the OpenStack cluster needs to be active all the time, thus we can only control the power states of the

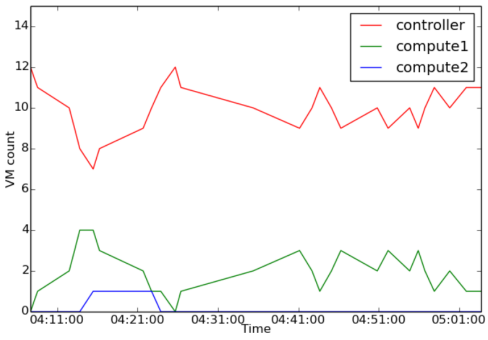

Fig. 7: VMs in the cluster during rate 0.4 test

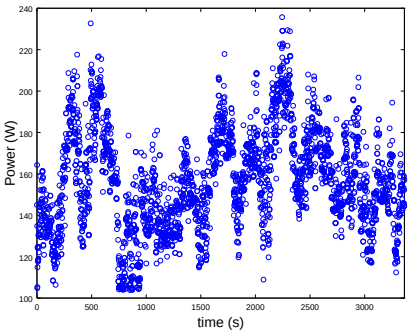

Fig. 8: Cluster power consumption during the rate 0.4 test

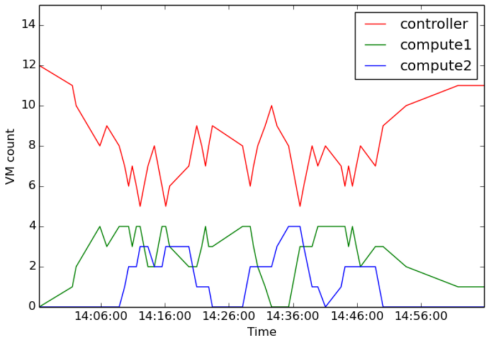

Fig. 9: VMs in the cluster during rate 0.6 test

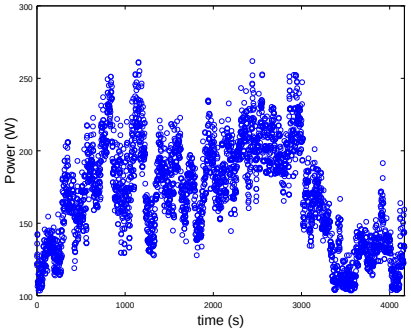

Fig. 10: Cluster power consumption during the rate 0.6 test

computing nodes. In Figure 11, we show the times that the computing nodes run without any virtual machines, i.e., idle and ready for energy state change, e.g. shutdown. In these values we have already taken into account the time it takes to switch from a state to another; boot up and shutdown. In the case of the test environment, the reboot time is approximately 76 seconds. So this has been deducted from the idle times as many times as there are state changes. In order to improve the performance and energy efficiency in our test environment, the boot times could be optimized with operating system and BIOS tuning or with faster hard drives.

Since the idle time does not necessarily indicate the potential energy savings alone, we estimated energy consumption based on the measured idle times shown in Figure 11. With these times and the idle energy consumption of the servers, 38W, we can calculated the energy saving estimates shown in Table 2. The energy savings potential, energy while idle, estimates how much energy could be saved by turning off the unnecessary servers.

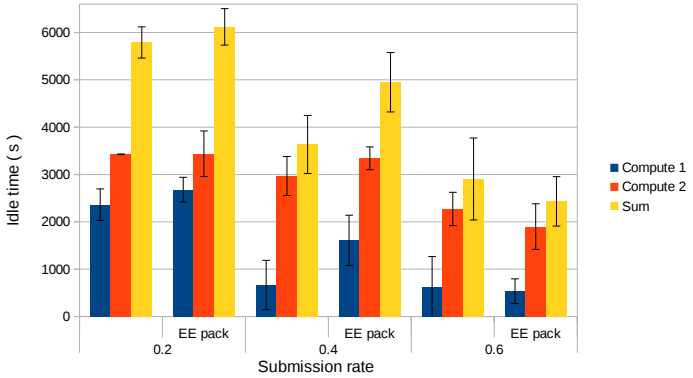

Fig. 11: Time the computing nodes are without virtual machines

Table 2: Energy saving potential (Wh) with different submission rates

Rate		0.2		0.4		0.6	
		Avg.	Stdev	Avg.	Stdev	Avg.	Stdev
NoEE		127.46	0.36	145.80	0.49	163.16	0.52
EE	total energy	125.68	0.31	145.99	1.33	181.13	15.12
	energy while idle	61.11	3.47	38.38	6.48	30.66	9.14
EE packing	total energy	126.06	1.25	148.73	9.64	170.16	6.50
	energy while idle	65.34	4.70	51.83	5.57	24.82	2.17

5.2 Simulation results

After the tests in the physical test cluster, the load balancing algorithm was tested in a simulator. The simulator allowed us to test with a larger server count and study the use of heterogeneous hardware.

We first tested how the virtual machines placement works in the simulator. In Figure 12, we show the server usage from the beginning of the simulation. It shows the number of free park servers and the number of free computing nodes. In the beginning, when the workloads have not started yet, the virtual machines are aggressively migrated to the park servers. As the workload increases, the number free execution nodes decreases. In the case of lower load, the need for execution nodes greatly varies. In the case of higher load, in Figure 13 the park servers are less important as the virtual machines are on the computing nodes.

Then we tested how energy efficiency of the cluster is affected by different threshold values in the second function of the load balancer, i.e., at which park server load level, the virtual machines are moved from the park server to computing nodes. In Figure 14 we can see the average energy consumption per instruction for different variations of the test. It also shows how the choice of park server hardware affects the energy consumption. Workload for these results is the same 100 jobs per hour per virtual machine. We can see how the energy efficiency improves as the load threshold and the number of park

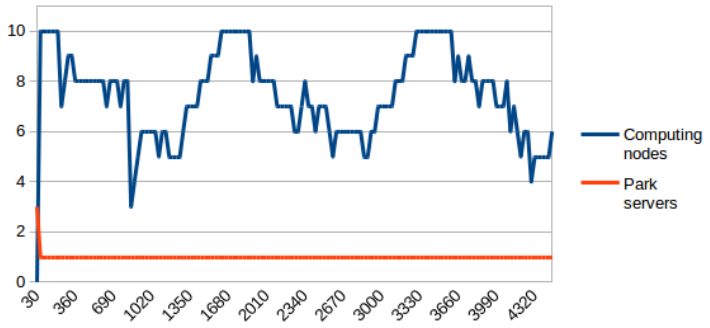

Fig. 12: Amount of idle computing nodes and park servers with rate 50 load (the x-axis is time)

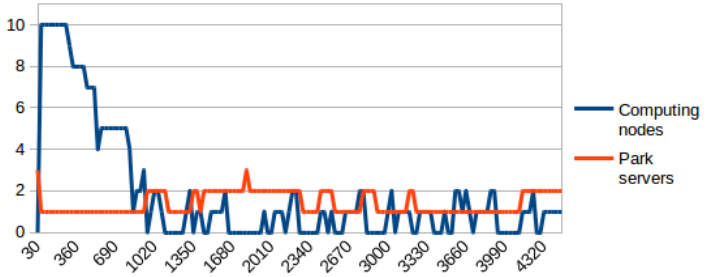

Fig. 13: Amount of idle computing nodes and park servers with rate 150 load (the x-axis is time)

servers are increased. An increased number of park servers allows the system to pack virtual machines more densely when there is less load.

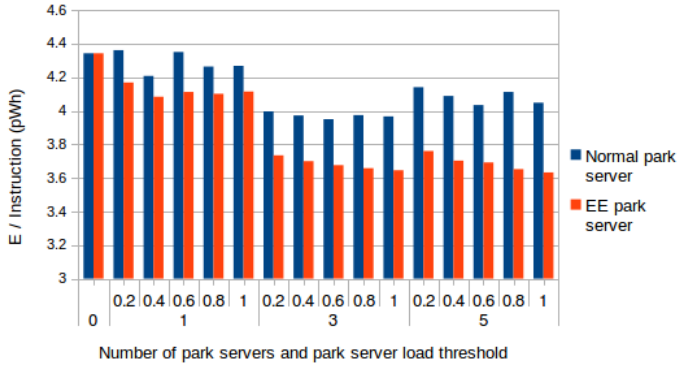

Fig. 14: Average energy consumption per instruction with rate 100 load

The improved energy efficiency has a drawback as the total amount of work and service level decreases. Figure 15 shows how the total amount of work, measured as the instruction count, is affected by the change of the park server threshold and the number of park servers. Similarly, Figure 16 shows how the service level decreases when the utilization level of park server increases. In this case, the SLA (service level agreements) describes the percentage of completed instructions, i.e., the service level.

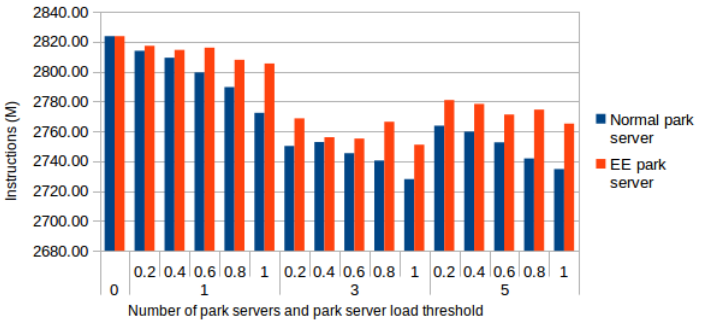

Fig. 15: Simulation time instructions with rate 100 load

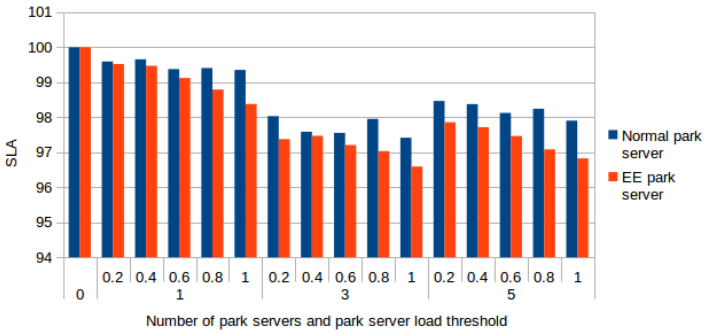

Fig. 16: SLA with rate 100 load

As seen earlier in Figures 12 and 13, the load of the system greatly affects the energy efficiency and the importance of load balancing to and from park servers. In Figure 17, we have similar results as earlier on the case of the rate 100 (Figure 14), but now with the higher load. The energy efficiency gets worse than it would without the load balancing functions as migrations introduce some overhead.

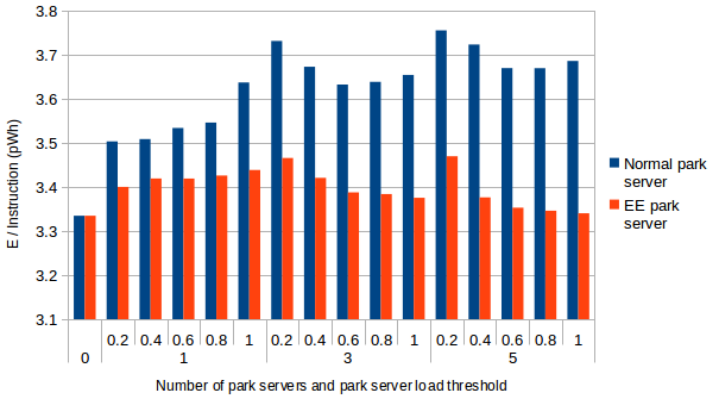

Fig. 17: Average energy consumption per instruction with rate 150 load

6 Conclusions and Future Work

The aim of this study was to show that the energy efficiency of high-energy physics computing could be improved in a cloud environment by using an active load based management of virtual machines. We introduced an idea of storing idle virtual machines on a dedicated hardware and load based over-committing of resources of idle virtual machines.

For this study we set up a cloud test environment with monitoring system and measured the effect of active management of virtual machines. We used well-known open source tools for the environment and realistic physics load. Our tests show that we can save 9% - 48% of energy when the virtual machines are managed actively based on the load of the system. The same algorithm was implemented into a simulator. Simulation results also show that the energy efficiency is improved by up to 40% with active management and with the use of heterogeneous hardware. The use of energy efficient hardware in park server improves the energy efficiency about 4.5% - 10%. The results also show, that the energy savings of the active management depends greatly on the workload.

In this study, we demonstrated that the dynamic management works with very basic heuristics. With a more sophisticated heuristics the results could be clearly better. For example, using load prediction methods [14], VMs could be moved earlier, or sometime a migration would be omitted if VM's load is predicted to increase soon again. We could also apply a fuzzy heuristics [32] to predict the memory consumption of VMs to avoid overloading of servers. It would also be possible to extend the model to handle more than two types of servers. This could improve efficiency, e.g. VMs running memory-intensive workloads could be place large memory servers. Also, the hardware setup in our tests was limited. Thus, the next steps would be to use a test system in a larger cloud environment. Also, load predictions such as the one by Ghosh et all [16] were not really considered in this study, but should be implemented into the logic of choosing migratable virtual machines. In the future testing,

also the load measurement should be reconsidered, since in the current tests with real hardware we used the load values given by Oracle Grid Engine and additionally the one minute load value given by the Linux kernel. It is possible that more real time load measurements and possible forecasting of future load, would clearly improve the load balancing function. Finally, an interesting future approach would be applying the same methods to containers [3,19]. Challenges can be related e.g. measuring their load and interactions between containers, since they are not so isolated as virtual machines.

Acknowledgement

This paper has received funding from the European Union's Horizon 2020 research and innovation program 2014-2018 under grant agreement No. 644866.

Disclaimer

This paper reflects only the authors' views and the European Commission is not responsible for any use that may be made of the information it contains.

References

1. Ahmed, A., Sabyasachi, A.S.: Cloud computing simulators: A detailed survey and future direction. In: Advance Computing Conference (IACC), 2014 IEEE International, pp. 866–872 (2014). DOI 10.1109/IAdCC.2014.6779436
2. Antcheva, I., Ballintijn, M., Bellenot, B., Biskup, M.: Root?a c++ framework for petabyte data storage, statistical analysis and visualization. Computer Physics Communications **180**(12), 2499?2512 (2009)
3. Banga, G., Druschel, P., Mogul, J.C.: Resource containers: A new facility for resource management in server systems. In: OSDI, vol. 99, pp. 45–58 (1999)
4. Barroso, L.A., Hölzle, U.: The case for energy-proportional computing. Computer **40**, 33–37 (2007)
5. Beloglazov, A., Buyya, R.: Energy efficient allocation of virtual machines in cloud data centers. In: Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on, pp. 577–578 (2010). DOI 10.1109/CCGRID.2010.45
6. Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr. Comput. : Pract. Exper. **24**(13), 1397–1420 (2012). DOI 10.1002/cpe.1867. URL http://dx.doi.org/10.1002/cpe.1867
7. Breitgand, D., Dubitzky, Z., Epstein, A., Glikson, A., Shapira, I.: Sla-aware resource over-commit in an iaas cloud. In: Proceedings of the 8th International Conference on Network and Service Management, CNSM '12, pp. 73–81. International Federation for Information Processing, Laxenburg, Austria, Austria (2013). URL http://dl.acm.org/citation.cfm?id=2499406.2499415
8. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience **41**(1), 23–50 (2011). DOI 10.1002/spe.995. URL http://dx.doi.org/10.1002/spe.995
9. Chung-Hsing, H., Poole, S.: Revisiting server energy proportionality. In: Parallel Processing (ICPP), 2013 42nd International Conference on, pp. 834–840 (2013). DOI 10.1109/ICPP.2013.99

10. Cioara, T., Anghel, I., Salomie, I., Copil, G., Moldovan, D., Kipp, A.: Energy aware dynamic resource consolidation algorithm for virtualized service centers based on reinforcement learning. In: Parallel and Distributed Computing (ISPDC), 2011 10th International Symposium on, pp. 163–169 (2011). DOI 10.1109/ISPDC.2011.32
11. Corradi, A., Fanelli, M., Foschini, L.: {VM} consolidation: A real case based on openstack cloud. Future Generation Computer Systems **32**(0), 118 – 127 (2014). DOI http://dx.doi.org/10.1016/j.future.2012.05.012. URL http://www.sciencedirect.com/science/article/pii/S0167739X12001082. Special Section: The Management of Cloud Systems, Special Section: Cyber-Physical Society and Special Section: Special Issue on Exploiting Semantic Technologies with Particularization on Linked Data over Grid and Cloud Architectures
12. Crago, S., Dunn, K., Eads, P., Hochstein, L., Kang, D.I., Kang, M., Modium, D., Singh, K., Suh, J., Walters, J.: Heterogeneous cloud computing. In: Cluster Computing (CLUSTER), 2011 IEEE International Conference on, pp. 378–385 (2011). DOI 10.1109/CLUSTER.2011.49
13. Dabbagh, M., Hamdaoui, B., Guizani, M., Rayes, A.: Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment. Network, IEEE **29**(2), 56–61 (2015). DOI 10.1109/MNET.2015.7064904
14. Dinda, P.A., O'Hallaron, D.R.: Host load prediction using linear models. Cluster Computing **3**(4), 265–280 (2000)
15. Fabozzi, F., Jones, C., Hegner, B., Lista, L.: Physics analysis tools for the cms experiment at lhc. Nuclear Science, IEEE Transactions on **55**, 3539–3543 (2008)
16. Ghosh, R., Naik, V.: Biting off safely more than you can chew: Predictive analytics for resource over-commit in iaas cloud. In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, pp. 25–32 (2012). DOI 10.1109/CLOUD.2012.131
17. Grzonka, D., Kolodziej, J., Tao, J., et al.: The analysis of openstack cloud computing platform: Features and performance. Journal of Telecommunications and Information Technology (3), 52 (2015)
18. Gupta, A., Milojicic, D., Kalé, L.V.: Optimizing vm placement for hpc in the cloud. In: Proceedings of the 2012 Workshop on Cloud Services, Federation, and the 8th Open Cirrus Summit, FederatedClouds '12, pp. 1–6. ACM, New York, NY, USA (2012). DOI 10.1145/2378975.2378977. URL http://doi.acm.org/10.1145/2378975.2378977
19. He, S., Guo, L., Guo, Y., Wu, C., Ghanem, M., Han, R.: Elastic application container: A lightweight approach for cloud resource provisioning. In: 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, pp. 15–22 (2012). DOI 10.1109/AINA.2012.74
20. Hermenier, F., Lorca, X., Menaud, J.M., Muller, G., Lawall, J.: Entropy: a consolidation manager for clusters. In: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, VEE '09, pp. 41–50. ACM, New York, NY, USA (2009). DOI 10.1145/1508293.1508300. URL http://doi.acm.org/10.1145/1508293.1508300
21. Hosseinimotlagh, S., Khunjush, F., Samadzadeh, R.: Seats: smart energy-aware task scheduling in real-time cloud computing. The Journal of Supercomputing **71**(1), 45–66 (2015). DOI 10.1007/s11227-014-1276-9. URL http://dx.doi.org/10.1007/s11227-014-1276-9
22. Intel: Your source for intel product specifications. Tech. rep., Intel Corporation (2016)
23. Jackson, K., Bunch, C., Sigler, E.: OpenStack cloud computing cookbook. Packt Publishing Ltd (2015)
24. Kommeri, J., Niemi, T., Helin, O.: Energy efficiency of server virtualization. International Journal On Advances in Intelligent Systems, v 5 n 3&4 (2012)
25. von Laszewski, G., Diaz, J., Wang, F., Fox, G.C.: Comparison of multiple cloud frameworks. In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, pp. 734–741 (2012). DOI 10.1109/CLOUD.2012.104
26. Li, B., Li, J., Huai, J., Wo, T., Li, Q., Zhong, L.: Enacloud: An energy-saving application live placement approach for cloud computing environments. In: Cloud Computing, 2009. CLOUD '09. IEEE International Conference on, pp. 17–24 (2009). DOI 10.1109/CLOUD.2009.72
27. Litvinski, O., Gherbi, A.: Experimental evaluation of openstack compute scheduler. Procedia Computer Science **19**(0), 116 – 123 (2013)

28. Medrano Llamas, R., Barreiro, M., Fernando, H., Kucharczyk, K., Denis, M.K., Cinquilli, M.: Commissioning the cern it agile infrastructure with experiment workloads. In: 20th International Conference on Computing in High Energy and Nuclear Physics 2013 (2013)
29. Meinhard, H.: Virtualization, clouds and iaas at cern. In: Proceedings of the 6th International Workshop on Virtualization Technologies in Distributed Computing Date, VTDC '12, pp. 27–28. ACM, New York, NY, USA (2012). DOI 10.1145/2287056.2287064. URL http://doi.acm.org/10.1145/2287056.2287064
30. Meusel, R., Blomer, J., Buncic, P., Ganis, G., Heikkilä, S.S.: Recent developments in the cernvm-file system server backend. Journal of Physics: Conference Series **608**(1), 012,031 (2015)
31. Nathuji, R., Schwan, K.: Virtualpower: coordinated power management in virtualized enterprise systems. SIGOPS Oper. Syst. Rev. **41**(6), 265–278 (2007). DOI 10.1145/1323293.1294287. URL http://doi.acm.org/10.1145/1323293.1294287
32. Niemi, T., Hameri, A.P.: Memory-based scheduling of scientific computing clusters. The Journal of Supercomputing **61**(3), 520–544 (2012)
33. Ou, Z., Zhuang, H., Nurminen, J.K., Ylä-Jääski, A., Hui, P.: Exploiting hardware heterogeneity within the same instance type of amazon ec2. In: Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Ccomputing, HotCloud'12, pp. 4–4. USENIX Association, Berkeley, CA, USA (2012). URL http://dl.acm.org/citation.cfm?id=2342763.2342767
34. Pahlavan, A., Momtazpour, M., Goudarzi, M.: Power reduction in hpc data centers: a joint server placement and chassis consolidation approach. The Journal of Supercomputing **70**(2), 845–879 (2014). DOI 10.1007/s11227-014-1265-z. URL http://dx.doi.org/10.1007/s11227-014-1265-z
35. Peng, J., Zhang, X., Lei, Z., Zhang, B., Zhang, W., Li, Q.: Comparison of several cloud computing platforms. In: 2009 Second International Symposium on Information Science and Engineering, pp. 23–27 (2009). DOI 10.1109/ISISE.2009.94
36. Ponce, S., Hersch, R.D.: Parallelization and scheduling of data intensive particle physics analysis jobs on clusters of pcs. In: 18th International Parallel and Distributed Processing Symposium (IPDPS 2004), CD-ROM / Abstracts Proceedings, 26-30 April 2004, Santa Fe, New Mexico, USA (2004). DOI 10.1109/IPDPS.2004.1303280. URL http://dx.doi.org/10.1109/IPDPS.2004.1303280
37. Ross, S.M.: Stochastic Processes, 2nd Edition. John Wiley and Sons (1996)
38. Sato, K., Samejima, M., Komoda, N.: Dynamic optimization of virtual machine placement by resource usage prediction. In: 2013 11th IEEE International Conference on Industrial Informatics (INDIN), pp. 86–91 (2013). DOI 10.1109/INDIN.2013.6622863
39. Sevalnev, M., Aalto, S., Kommeri, J., Niemi, T.: Using queuing theory for controlling the number of computing servers. In: ICGREEN 2012 (Third International Conference on Green IT Solutions (2012)
40. Sharifi, M., Salimi, H., Najafzadeh, M.: Power-efficient distributed scheduling of virtual machines using workload-aware consolidation techniques. J. Supercomput. **61**(1), 46–66 (2012). DOI 10.1007/s11227-011-0658-5. URL http://dx.doi.org/10.1007/s11227-011-0658-5
41. Srikantaiah, S., Kansal, A., Zhao, F.: Energy aware consolidation for cloud computing. In: Proceedings of the 2008 conference on Power aware computing and systems, HotPower'08, pp. 10–10. USENIX Association, Berkeley, CA, USA (2008). URL http://dl.acm.org/citation.cfm?id=1855610.1855620
42. Takahiro, H., Hidemoto, N., Satoshi, I., Satoshi, S.: Reactive cloud: Consolidating virtual machines with postcopy live migration. Information and Media Technologies **7**(2), 614–626 (2012)
43. Takouna, I., Meinel, C.: Coordinating vms' memory demand heterogeneity and memory dvfs for energy-efficient vms consolidation. In: Internet of Things (iThings), 2014 IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE and Cyber, Physical and Social Computing(CPSCom), IEEE, pp. 478–485 (2014). DOI 10.1109/iThings.2014.85
44. Tomás, L., Tordsson, J.: Improving cloud infrastructure utilization through overbooking. In: Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference, CAC

'13, pp. 5:1–5:10. ACM, New York, NY, USA (2013). DOI 10.1145/2494621.2494627. URL http://doi.acm.org/10.1145/2494621.2494627
45. Verma, A., Ahuja, P., Neogi, A.: Power-aware dynamic placement of hpc applications. In: Proceedings of the 22nd annual international conference on Supercomputing, ICS '08, pp. 175–184. ACM, New York, NY, USA (2008)
46. Wang, X., Liu, X., Fan, L., Jia, X.: A decentralized virtual machine migration approach of data centers for cloud computing. Mathematical Problems in Engineering p. 10 (2013)
47. Wen, X., Gu, G., Li, Q., Gao, Y., Zhang, X.: Comparison of open-source cloud management platforms: Openstack and opennebula. In: Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on, pp. 2457–2461 (2012). DOI 10.1109/FSKD.2012.6234218
48. Wuhib, F., Stadler, R., Lindgren, H.: Dynamic resource allocation with management objective: Implementation for an openstack cloud. In: Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm), pp. 309–315 (2012)
49. Younge, A., Fox, G.: Advanced virtualization techniques for high performance cloud cyberinfrastructure. In: Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, pp. 583–586 (2014). DOI 10.1109/CCGrid.2014.93